

CLAIMS

Sec 17
1. A method for managing flow of messages between two software modules, said method comprising:

5 (a) determining whether a first message can be propagated by a first thread running on a first processor between the two software modules while allowing a second thread running on a second processor to propagate a second message between the two software modules; and

10 (b) propagating the first message between the two software modules while allowing the second thread to propagate the second message between the two software modules when said determining (a) determines that the first message can be propagated by the first thread while allowing the second thread to propagate the second message between the two software modules.

15 2. A method as recited in claim 1, wherein the first and second threads concurrently propagate respective portions of the first and second messages between the two software modules.

3. A method as recited in claim 1, wherein the method further comprises:

20 (c) blocking the second thread from propagating the second message between the two software modules when said determining (a) determines that the first message cannot be propagated by the first thread while allowing the second message to propagate between the two layer software modules.

4. A method as recited in claim 2, wherein said blocking (c) of the second thread from propagating the message between the two software modules is achieved by providing a lock which can be acquired by the first thread.

Sec 2
25 5. A method as recited in claim 1, wherein the method further comprises:

setting a first indicator for the first processor to indicate that the first processor is propagating when said determining (a) determines that the first

message can be propagated by the first thread while allowing the second thread to propagate the second message between the two layer software modules.

setting the first indicator for the first processor to indicate that the first processor is not propagating when said propagating (b) has propagated the first message between the two software modules.

6. A method as recited in claim 1, wherein said determining (a) comprises:

(a1) determining whether an event is being processed or is pending to be processed; and

(a2) determining whether a thread-count for the first processor is zero, and

wherein said determining (a) determines that the first thread can propagate the first message without blocking the second thread from propagating the second message when said determining (a) determines that no events is being processed or pending and said determining (a) determines that the thread-count for the first processor is zero.

7. A method as recited in claim 1, wherein the method further comprises:

(a1) determining whether a synchronization queue associated with one of the software modules contains one or more data messages.

8. A method as recited in claim 6, wherein the method further comprises:

propagating the first message to the bottom of the synchronization queue when the determining (a1) determines that the synchronization queue contains one or more messages; and

propagating a message from the head of the synchronization queue when the determining determines (a1) that the synchronization queue contains one or more messages.

9. A method as recited in claim 6, wherein the method further comprises:

propagating the first message to the bottom of a queue of one of the software modules when the determining determines (a1) that the synchronization queue does not contain one or more messages.

5 10. A method as recited in claim 1, wherein the two software modules are implemented in a stack as STREAMS modules.

11. A computer system comprising:

a plurality of processors;

first and second software modules, the second software module having a
10 main queue suitable for storing messages and an auxiliary queue suitable for storing messages that are not stored in the main queue; and

a propagation controller operating to enable at least two processors of said plurality of processors to concurrently propagate messages between the first and the second software modules.

15 12. A computer system as recited in claim 11,

wherein the propagation controller comprises:

a thread-count for one of said plurality of processors; and

a queue count for the auxiliary queue.

13. A computer system as recited in claim 12,

20 wherein the auxiliary queue is a synchronization queue and the queue count is a synchronization queue count.

14. A computer system as recited in claim 11, wherein the at least two processors of said plurality of processors concurrently propagate a message from the first software module to the auxiliary queue of the second software module.

15. A computer system as recited in claim 11, wherein the at least two processors of said plurality of processors concurrently propagate a message from the first software module to the main queue of the second software module.

16. A computer system as recited in claim 16, wherein the first and second software modules are implemented in a stack as STREAMS modules.

17. A computer readable media including computer program code for managing flow of messages between two software modules, said computer readable media comprising:

computer program code for determining whether a first message can be propagated by a first thread running on a first processor between the two software modules while allowing a second thread running on a second processor to propagate a second message between the two software modules; and

computer program code for propagating the first message between the two software modules while allowing the second thread to propagate the second message between the two software modules when said computer program code for determining determines that the first message can be propagated by the first thread while allowing the second thread to propagate the second message between the two software modules.

18. A computer readable media as recited in claim 17, wherein the first and second threads concurrently propagate respective portions of the first and second messages between the two software modules.

19. A computer readable media as recited in claim 18, wherein the computer readable media further comprises:

computer program code for setting a first indicator for the first processor to indicate that the first processor is propagating when said determining (a) determines that the first message can be propagated by the first thread while

and thread to propagate the second message to the STREAMS modules.

program code for setting the first indicator for the first processor is not propagating when said first message between the two software modules is received on a readable media as recited in claim 19, where the first message is implemented in a stack as STREAMS modules.

5

add
a5

add
a5)

~~Handwritten signature~~[illegible]